

The denoising Convolutional Autoencoder

Vision and Cognitive Systems - Lamberto Ballan

M. Giunta,

L. Rinaldi.

September 27, 2022



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

- 1 Introduction
- 2 Related Work
- 3 Dataset
- 4 Methods
 - Base Architecture
 - Improved Architectures
 - Loss Functions
- 5 Experiments
 - Architecture Experiments
 - Loss Function Experiments
 - Pixelation Experiments
- 6 Conclusions

- 1 Introduction
- 2 Related Work
- 3 Dataset
- 4 Methods
 - Base Architecture
 - Improved Architectures
 - Loss Functions
- 5 Experiments
 - Architecture Experiments
 - Loss Function Experiments
 - Pixelation Experiments
- 6 Conclusions

Privacy-preserving image corruption



When divulging sensitive photographic media it's common practice to artificially corrupt images to protect the privacy of the depicted subjects. (images from [10])



Examples: pixelation/blurring of police reports, videos and photos in public spaces, posting pictures of minors.

Many studies try to reverse blurring/pixelation with deep neural networks. There are many applications of denoising technologies:

Many studies try to reverse blurring/pixelation with deep neural networks. There are many applications of denoising technologies:

- Security: are these common privacy-preserving techniques really secure? Should we use more robust ones?

Many studies try to reverse blurring/pixelation with deep neural networks. There are many applications of denoising technologies:

- Security: are these common privacy-preserving techniques really secure? Should we use more robust ones?
- Photo editing: remove blurring from out of focus pictures.

Many studies try to reverse blurring/pixelation with deep neural networks. There are many applications of denoising technologies:

- Security: are these common privacy-preserving techniques really secure? Should we use more robust ones?
- Photo editing: remove blurring from out of focus pictures.
- Video streaming: counteract the loss of information caused by the compression and/or the noise added by video streaming services.

Many studies try to reverse blurring/pixelation with deep neural networks. There are many applications of denoising technologies:

- Security: are these common privacy-preserving techniques really secure? Should we use more robust ones?
- Photo editing: remove blurring from out of focus pictures.
- Video streaming: counteract the loss of information caused by the compression and/or the noise added by video streaming services.

For reasons like these it's interesting to build a DL-based denoiser; in particular we were interested in what's possible using limited computational resources. To do so we experimented with (variations of) the *Convolutional Autoencoder* architecture, three loss functions, and a few common performance metrics.

- 1 Introduction
- 2 Related Work**
- 3 Dataset
- 4 Methods
 - Base Architecture
 - Improved Architectures
 - Loss Functions
- 5 Experiments
 - Architecture Experiments
 - Loss Function Experiments
 - Pixelation Experiments
- 6 Conclusions

- *Convolutional neural networks* (CNN) are a deep neural network architecture commonly used for object recognition and image classification, known to yield high prediction accuracy in supervised learning tasks.

- *Convolutional neural networks* (CNN) are a deep neural network architecture commonly used for object recognition and image classification, known to yield high prediction accuracy in supervised learning tasks.
- Their effectiveness when dealing with images stems from their ability to capture multiple local features (via the multi-channel feature maps) while preserving the original spatial structure of the image.

- *Convolutional neural networks* (CNN) are a deep neural network architecture commonly used for object recognition and image classification, known to yield high prediction accuracy in supervised learning tasks.
- Their effectiveness when dealing with images stems from their ability to capture multiple local features (via the multi-channel feature maps) while preserving the original spatial structure of the image.
- This makes them useful in many facial recognition tasks; for example CNNs have been proved to be able to reach near 100% accuracy level in the classification task applied to the LFW dataset [15] [11].

- *Convolutional neural networks* (CNN) are a deep neural network architecture commonly used for object recognition and image classification, known to yield high prediction accuracy in supervised learning tasks.
- Their effectiveness when dealing with images stems from their ability to capture multiple local features (via the multi-channel feature maps) while preserving the original spatial structure of the image.
- This makes them useful in many facial recognition tasks; for example CNNs have been proved to be able to reach near 100% accuracy level in the classification task applied to the LFW dataset [15] [11].
- CNNs are known to be very accurate in classifying faces even if they have been previously blurred or pixelated [10]; this makes them the natural candidate for building a simple but effective denoiser.

Having shown why convolutional layers are reasonable building blocks we now discuss why it makes sense to use an *autoencoder* as the overall architecture.

- The autoencoder is a generative architecture, whose applications range from dimensionality reduction/feature learning to facial recognition [2].

Having shown why convolutional layers are reasonable building blocks we now discuss why it makes sense to use an *autoencoder* as the overall architecture.

- The autoencoder is a generative architecture, whose applications range from dimensionality reduction/feature learning to facial recognition [2].
- The idea behind a denoising autoencoder is to perform “smart compression”: we map the input to a lower dimensional space, thus forcing some information to be discarded; if learning is successful only the irrelevant noise will be thrown away, resulting in a “cleansed” image [2].

- In practice this means sending the images through an *encoder* made of several layers (which we choose to be convolutional for the above reasons); then the smaller dimensional representation goes through a *decoder*, which is a specular copy of the encoder. The result will be an image with the same resolution as the input.

- In practice this means sending the images through an *encoder* made of several layers (which we choose to be convolutional for the above reasons); then the smaller dimensional representation goes through a *decoder*, which is a specular copy of the encoder. The result will be an image with the same resolution as the input.
- This *convolutional autoencoder* (CAE) architecture is known to work well in facial recognition classification tasks [3] as well as denoising ones [13]; for these reasons we based our CAE architecture on [13], where the authors also aim to denoise blurred/pixelated images from the LFW dataset.

In addition to a base CAE architecture we also develop three improved versions:

- The *variational convolutional autoencoder* (VCAE), where we inject extra stochasticity at the latent space level. This has been shown to sometimes be able to boost performance in the denoising task [6].

In addition to a base CAE architecture we also develop three improved versions:

- The *variational convolutional autoencoder* (VCAE), where we inject extra stochasticity at the latent space level. This has been shown to sometimes be able to boost performance in the denoising task [6].
- The CAE with *residual blocks* (ResBlocks), which can also improve model performance [8].

In addition to a base CAE architecture we also develop three improved versions:

- The *variational convolutional autoencoder* (VCAE), where we inject extra stochasticity at the latent space level. This has been shown to sometimes be able to boost performance in the denoising task [6].
- The CAE with *residual blocks* (ResBlocks), which can also improve model performance [8].
- The base CAE but trained with one of two versions of the *Perceptual Loss* based on VGG16, which has been shown to be able to significantly improve results both in a denoising setting [13] as well as in a super-resolution one [7].

- To quantify model performance we use both the standard *peak signal to noise ratio* (PSNR) as well as the more sophisticated *structural similarity index measure* (SSIM) [14].
- At any case we remark that the final performance index must be human aesthetics, as these metrics are known to not correlate well with human perception of result quality; several studies show that that PSNR in particular is often outperformed by other visual quality metrics [9] [12] [1] [5], and we expect this to be the case in our problem too [13].

- 1 Introduction
- 2 Related Work
- 3 Dataset**
- 4 Methods
 - Base Architecture
 - Improved Architectures
 - Loss Functions
- 5 Experiments
 - Architecture Experiments
 - Loss Function Experiments
 - Pixelation Experiments
- 6 Conclusions

We used the *Labeled Faces in the Wild* [4] dataset to train our networks; this is a collection of 13,233 250×250 images of faces of celebrities collected from the web.

We used the *Labeled Faces in the Wild* [4] dataset to train our networks; this is a collection of 13,233 250×250 images of faces of celebrities collected from the web.

In particular we used the *funneled version* [4], that adds some preprocessing:

- First the faces are detected and centered using the openCV implementation of the Viola-Jones face detector;
- then images are cropped and enlarged by a factor of 2.2, so that they're now 224×224 images.
- These measures ensure heads are fully captured in the image and at an almost uniform size.

We then added some further custom preprocessing:

- First we perform a center crop (resulting in 112×112 images); this is done to ignore most of the backgrounds, since we want to focus on the facial region for the obfuscation task.
- Secondly we map images from the $[0, 255]$ RGB interval to the $[0, 1]$ interval (standard practice to improve neural network convergence);
- finally we normalize the images by adjusting the mean and the std so that they both equal to 0.5. This is because we noticed empirically that it helped accelerate convergence.

The last step was randomly dividing the dataset in 3 parts as follows:

- test set: 3708 samples, i.e. 30% of overall set;
- validation set: 1905 samples, i.e. 20% of the non-test images;
- training set: 7620 samples, i.e. 80% of the non-test images.

- To obfuscate the images to be passed as input to the network we used *gaussian blurring* for most of the time.

- To obfuscate the images to be passed as input to the network we used *gaussian blurring* for most of the time.
- This means convolving the 2D matrix specified by each image with a gaussian kernel whose size and variance parameter must be specified by the user; this results in an image where finer details are removed.

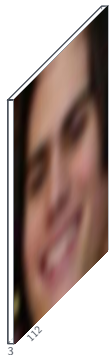
- To obfuscate the images to be passed as input to the network we used *gaussian blurring* for most of the time.
- This means convolving the 2D matrix specified by each image with a gaussian kernel whose size and variance parameter must be specified by the user; this results in an image where finer details are removed.
- In our experiments we used a gaussian kernel with size 9 and $\sigma = 4$, as these values represent a good compromise: the resulting obfuscated images are corrupted enough that facial features appear to be lost to the human observer, but not to the point that a simple CAE with reasonably limited training resources wouldn't be able to achieve good enough performance.

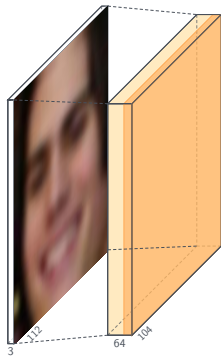
- The second obfuscation method we employed is *pixelation* a.k.a. *mosaicing*.

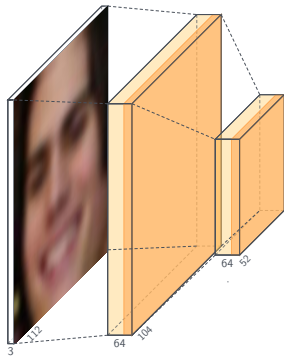
- The second obfuscation method we employed is *pixelation* a.k.a. *mosaicing*.
- This consists in dividing each image into $n \times n$ squares, which are then painted with the average color of the pixels in that square.

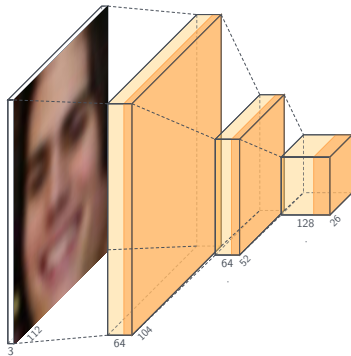
- The second obfuscation method we employed is *pixelation* a.k.a. *mosaicing*.
- This consists in dividing each image into $n \times n$ squares, which are then painted with the average color of the pixels in that square.
- We chose $n = 8$ to stress test the capabilities of the CAE architecture, as this is a more aggressive way of obfuscating the image - while still being gentle enough that effective reconstructions may be possible.

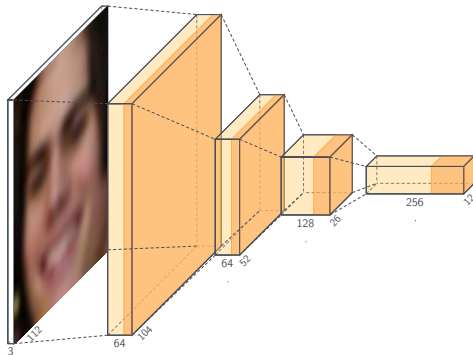
- 1 Introduction
- 2 Related Work
- 3 Dataset
- 4 Methods**
 - Base Architecture
 - Improved Architectures
 - Loss Functions
- 5 Experiments
 - Architecture Experiments
 - Loss Function Experiments
 - Pixelation Experiments
- 6 Conclusions

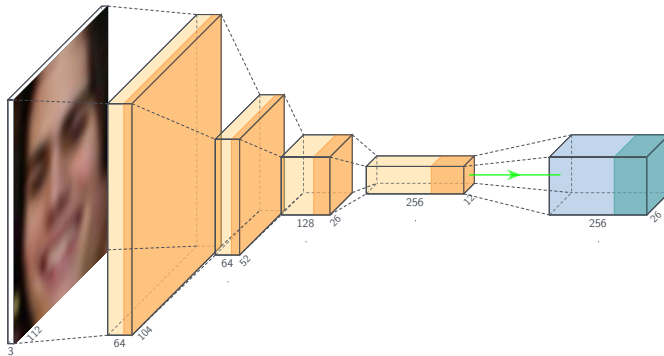




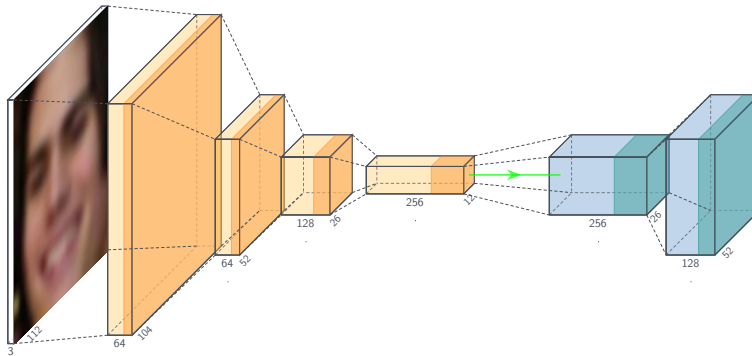


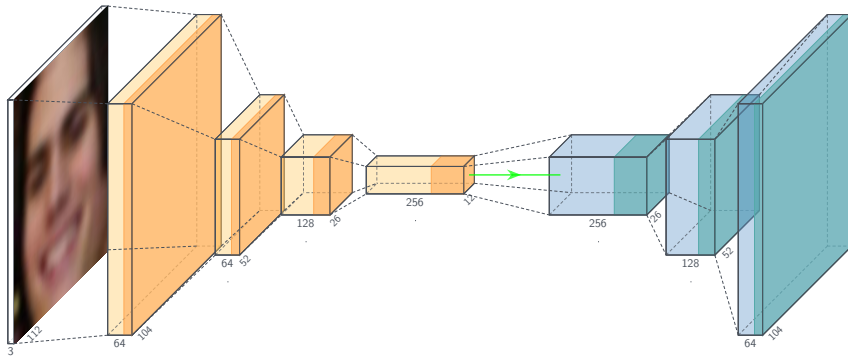


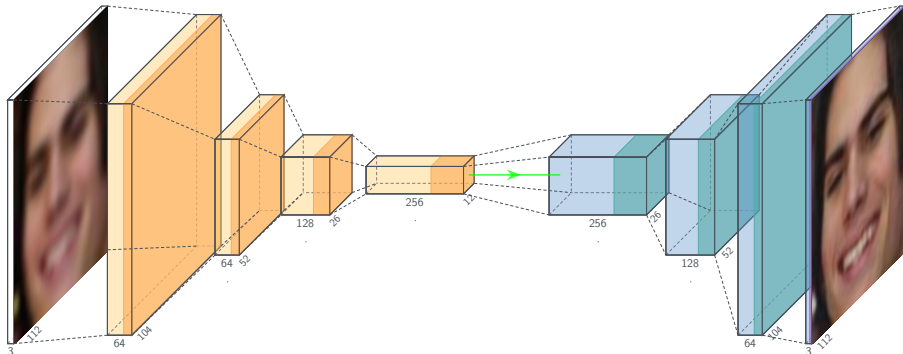




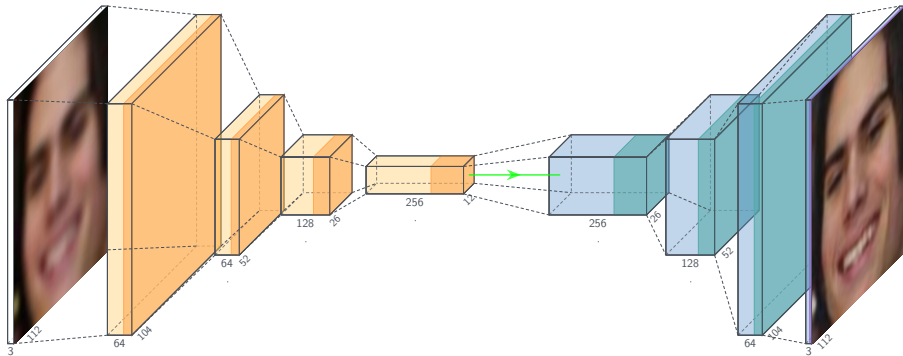
Base Architecture

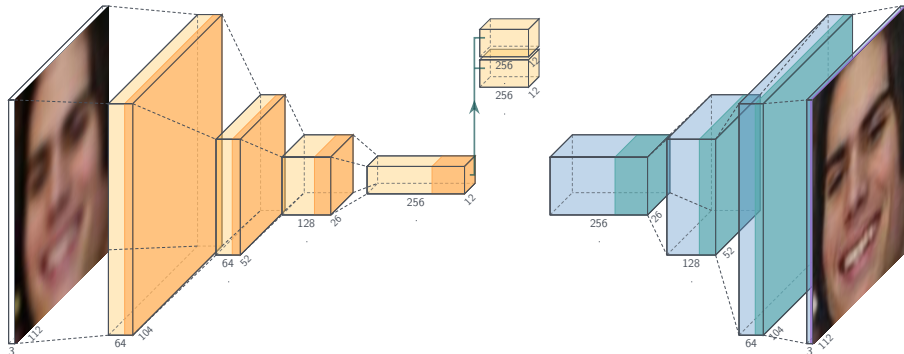


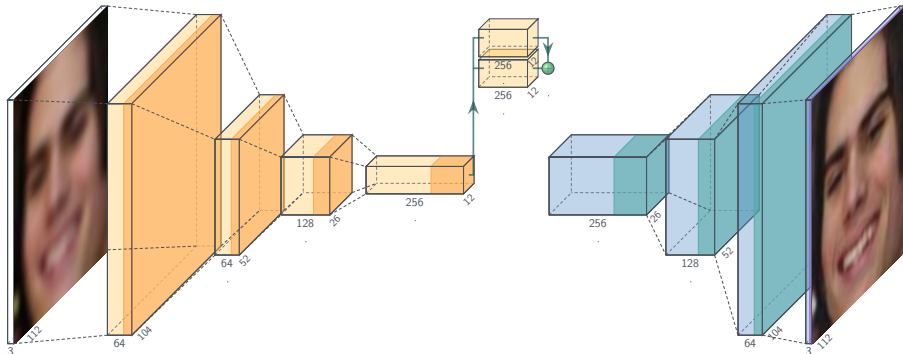


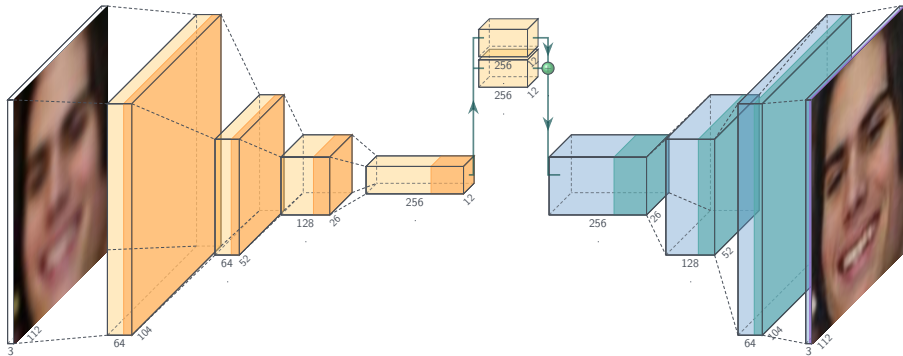


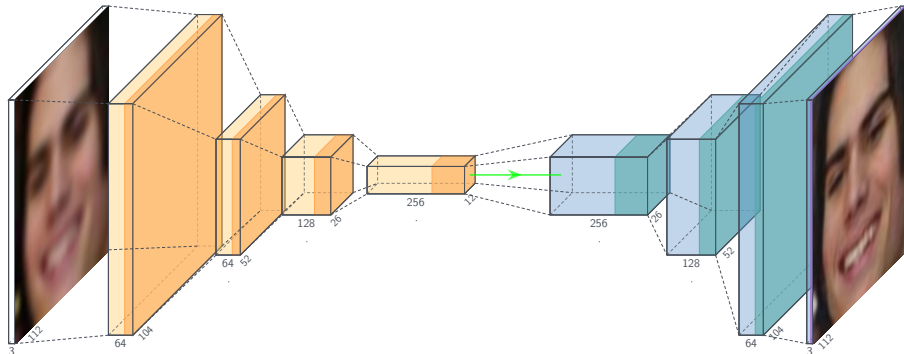
Variational CAE

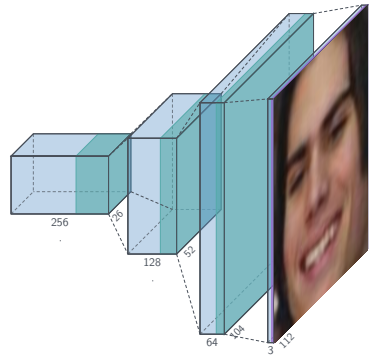
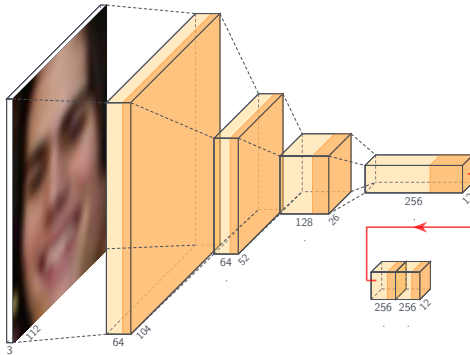


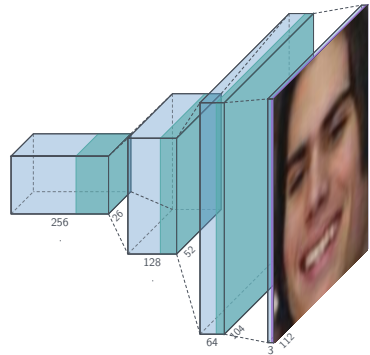
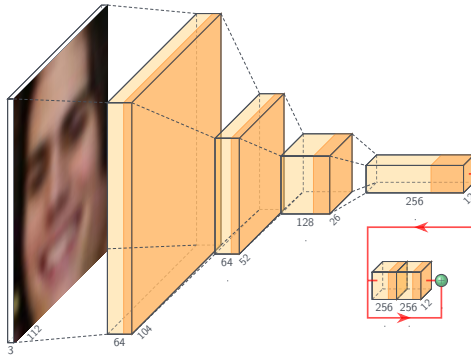


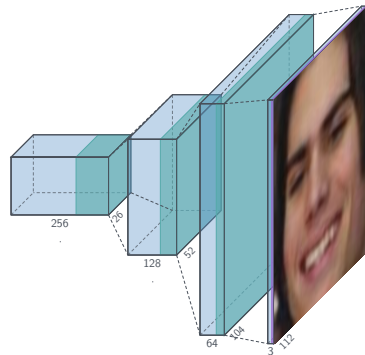
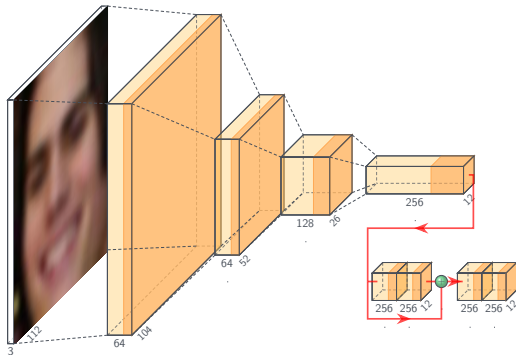


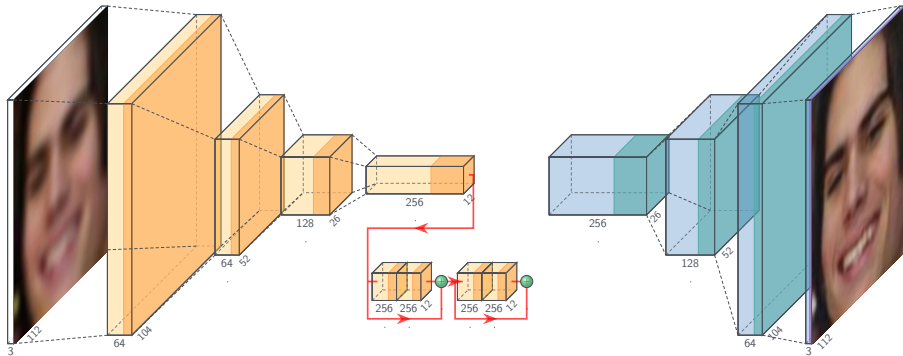


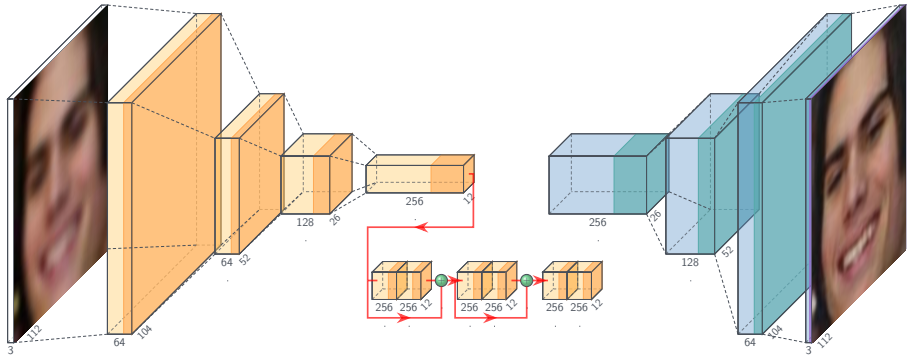


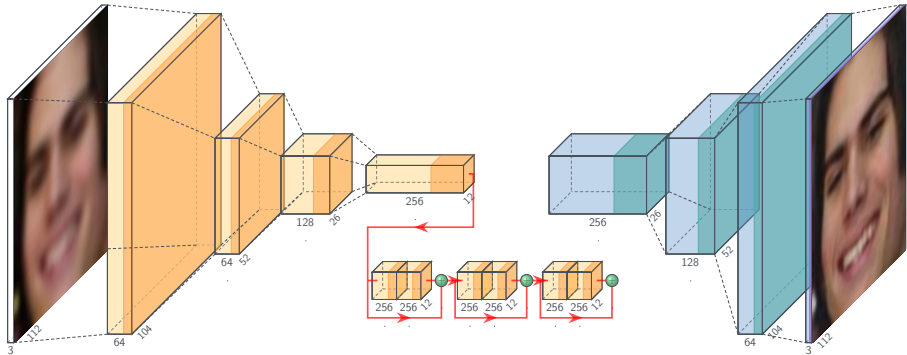


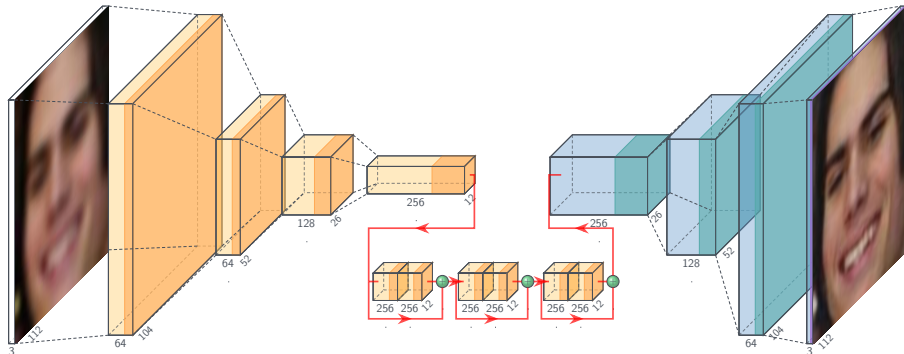












The first type of loss we implemented is the *pixel loss*, which is the MSE between the original and the reconstructed image:

The first type of loss we implemented is the *pixel loss*, which is the MSE between the original and the reconstructed image:

$$\ell_{\text{pixel}}(x, \hat{x}) = \frac{1}{112 \times 112 \times 3} \|x - \hat{x}\|_2^2$$

The first type of loss we implemented is the *pixel loss*, which is the MSE between the original and the reconstructed image:

$$\ell_{\text{pixel}}(x, \hat{x}) = \frac{1}{112 \times 112 \times 3} \|x - \hat{x}\|_2^2$$

where the $112 \times 112 \times 3$ is simply the number of pixels in our RGB images.

The pixel loss is a common choice with generative models dealing with pictures, and it make sense since it penalize the pixel-wise differences between the two images.

The pixel loss is a common choice with generative models dealing with pictures, and it make sense since it penalize the pixel-wise differences between the two images.

Still it has the issue of being “naive”: images that look similar to a human observer can be made to have relatively large pixel losses by having many small element-wise differences, and similarly relatively low losses can correspond to images that a human observer could clearly tell apart [13].

A loss that minimizes pixel differences captures low level features; a better way to approach the problem may be to look for one that minimizes differences between high level features.

A loss that minimizes pixel differences captures low level features; a better way to approach the problem may be to look for one that minimizes differences between high level features.

For example if the facial features are correctly reproduced with slightly different colors a human observer would say the result is good, even though the individual pixels may be significantly different. This can be achieved using transfer learning, to exploit previously gained knowledge about these high level features (e.g. the shape of the eyes, hair, etc.).

In this work we use the *perceptual loss* implemented in [7], defined as follows:

$$\ell_{\text{perceptual}}(x, \hat{x}) = \frac{1}{H \times W \times C} \|\varphi(x) - \varphi(\hat{x})\|_2^2$$

The perceptual loss is the MSE computed between the feature representations $\varphi(x)$, $\varphi(\hat{x})$ of the two images, whose shapes are (H, W, C) . Following [7] we define φ as the output of the `relu2_2` layer of the VGG16 network pretrained on ImageNet with its weights fixed.

VGG16's expected input has a shape equal to $(224, 224, 3)$, so in order to implement our perceptual loss we added an upsampling layer with linear interpolation to double the length size.

VGG16's expected input has a shape equal to $(224, 224, 3)$, so in order to implement our perceptual loss we added an upsampling layer with linear interpolation to double the length size.

The combination of this upsampling layer, the first 15 VGG16 layers, and finally the MSE constitutes the loss network we used to better capture the differences in content features in the images.

VGG16's expected input has a shape equal to $(224, 224, 3)$, so in order to implement our perceptual loss we added an upsampling layer with linear interpolation to double the length size.

The combination of this upsampling layer, the first 15 VGG16 layers, and finally the MSE constitutes the loss network we used to better capture the differences in content features in the images.

Finally notice that this feature representation has shape $(H = 56, W = 56, C = 256)$.

In order to try to capture both high and low level features we tested a *mixed perceptual loss* given by:

In order to try to capture both high and low level features we tested a *mixed perceptual loss* given by:

$$\ell_{\text{mixed}} = \rho \ell_{\text{pixel}} + \ell_{\text{perceptual}}$$

In order to try to capture both high and low level features we tested a *mixed perceptual loss* given by:

$$\ell_{\text{mixed}} = \rho \ell_{\text{pixel}} + \ell_{\text{perceptual}}$$

The ρ parameter quantifies the relative importance of each contribution: for example if $\rho = 1$ the two terms are weighed equally, whereas if $\rho = 0.5$ the perceptual loss is twice as important as the pixel one.

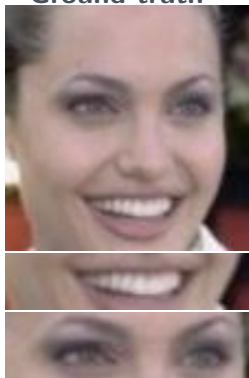
- 1 Introduction
- 2 Related Work
- 3 Dataset
- 4 Methods
 - Base Architecture
 - Improved Architectures
 - Loss Functions
- 5 Experiments**
 - Architecture Experiments
 - Loss Function Experiments
 - Pixelation Experiments
- 6 Conclusions

At first we performed a small-scope grid search to choose the hyperparameters for the CAE that optimize the PSNR / SSIM scores; this allowed us to choose a batch size of 8, an Adam optimizer with an initial LR of $5 \cdot 10^{-4}$, and a small L2 regularization with a 10^{-5} strength.

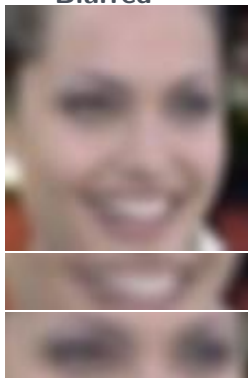
At first we performed a small-scope grid search to choose the hyperparameters for the CAE that optimize the PSNR / SSIM scores; this allowed us to choose a batch size of 8, an Adam optimizer with an initial LR of $5 \cdot 10^{-4}$, and a small L2 regularization with a 10^{-5} strength.

We also chose to run training for up to 15 epochs, as this was a good compromise between performance (the loss always flattened at around 15 epochs, and often stopped significantly improving after 5-10 epochs) and training time (using between 5 and 15 epochs we were able to keep each training session under about an hour with the available GPUs). (cfr 1)

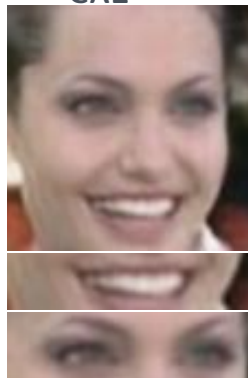
Ground truth



Blurred



CAE



This image
Test Set

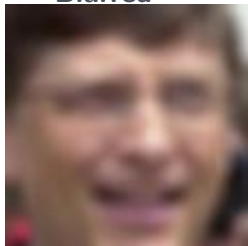
27.01 / 0.75
26.67 / 0.73

31.09 / 0.89
30.62 / 0.88

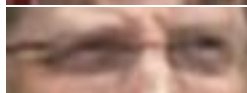
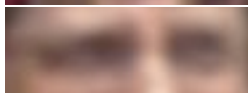
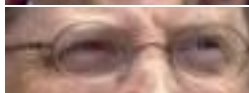
Ground truth



Blurred



CAE




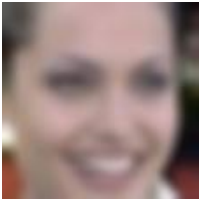

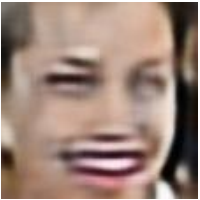




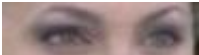
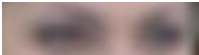
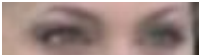
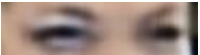
This image
Test Set

25.42 / 0.63
26.67 / 0.73

29.36 / 0.82
30.62 / 0.88

Testing Variational CAE


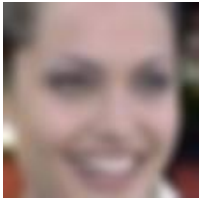

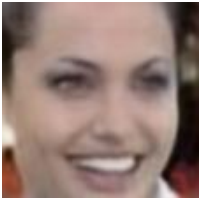
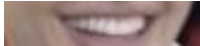

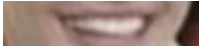
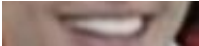
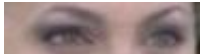
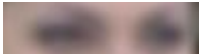
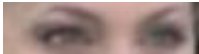
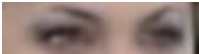


Ground truth	Blurred	CAE	VCAE
			
			
			
This image	27.01 / 0.75	31.09 / 0.89	20.47 / 0.53
Test Set	26.67 / 0.73	30.62 / 0.88	22.69 / 0.54

Testing Variational CAE



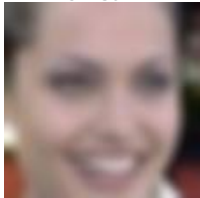
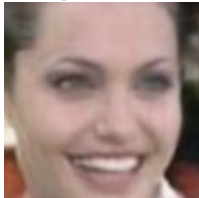


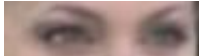
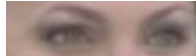
Ground truth	Blurred	CAE	VCAE
			
			
This image	25.42 / 0.63	29.36 / 0.82	21.20 / 0.49
Test Set	26.67 / 0.73	30.62 / 0.88	22.69 / 0.54

Ground truth	Blurred	CAE	ResCAE
			
			
			
This image	27.01 / 0.75	31.09 / 0.89	28.73 / 0.82
Test Set	26.67 / 0.73	30.62 / 0.88	29.43 / 0.84

Ground truth	Blurred	CAE	ResCAE
			
			
This image	25.42 / 0.63	29.36 / 0.82	28.73 / 0.82
Test Set	26.67 / 0.73	30.62 / 0.88	29.43 / 0.84


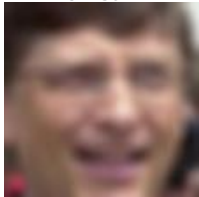


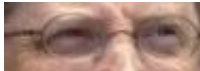
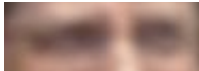
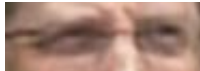
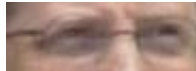
Testing Perceptual Loss



Ground truth	Blurred	CAE	CAE with PL
			
			
			
This image	27.01 / 0.75	31.09 / 0.89	27.97 / 0.85
Test Set	26.67 / 0.73	30.62 / 0.88	24.25 / 0.79









Testing Perceptual Loss



Ground truth	Blurred	CAE	CAE with PL
			
			
This image	25.42 / 0.63	29.36 / 0.82	24.24 / 0.74
Test Set	26.67 / 0.73	30.62 / 0.88	24.25 / 0.79


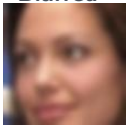
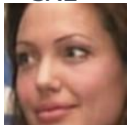



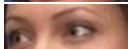

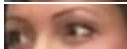
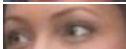
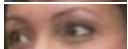
Improved convergence speed (perc. loss)



	epoch 1	epoch 5	epoch 10	epoch 15
Pixel loss	 24.04 / 0.62 23.19 / 0.62	 26.82 / 0.76 28.76 / 0.83	 26.51 / 0.75 29.72 / 0.85	 27.49 / 0.78 31.07 / 0.87
Perc. loss	 11.52 / 0.21 10.47 / 0.00	 20.07 / 0.50 18.27 / 0.45	 24.67 / 0.65 21.94 / 0.64	 26.55 / 0.72 24.28 / 31.4





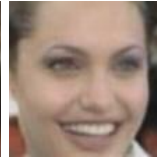



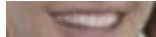
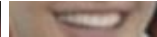
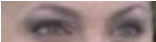
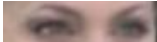
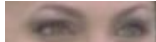
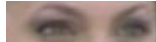
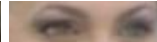
Improving color convergence (mixed loss)



	Base	Blurred	CAE	PL	MPL 0.5	MPL 1
						
						
This image		27.45/0.78	31.62/0.91	25.46/0.83	27.90/0.88	29.01/0.91
Test Set		26.67/0.73	30.62/0.88	24.25/0.79	26.59/0.84	27.26/0.86






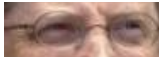

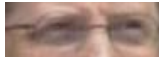
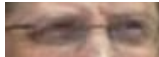
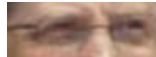
Testing Mixed Perceptual Loss



Ground truth	CAE	CAE with PL	MPL 0.5	MPL 1
				
				
				
This image	31.09 / 0.89	27.97 / 0.85	28.62 / 0.85	28.21 / 0.82
Test Set	30.62 / 0.88	24.25 / 0.79	26.59 / 0.84	27.26 / 0.86

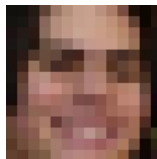
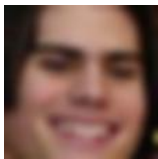
Testing Mixed Perceptual Loss



Ground truth	CAE	CAE with PL	MPL0.5	MPL 1
				
				
This image	29.36 / 0.82	24.24 / 0.74	27.48 / 0.81	27.32 / 0.80
Test Set	30.62 / 0.88	24.25 / 0.79	26.59 / 0.84	27.26 / 0.86

To push the limits of the denoising CAE we tried repeating the training on a smaller scale with a more aggressive corruption of the input images, via *pixelation* (*mosaicing*).

To push the limits of the denoising CAE we tried repeating the training on a smaller scale with a more aggressive corruption of the input images, via *pixelation* (*mosaicing*).



To push the limits of the denoising CAE we tried repeating the training on a smaller scale with a more aggressive corruption of the input images, via *pixelation* (*mosaicing*).

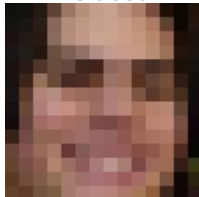


We notice that compared to the blurred dataset here more details are removed, resulting in an image with less details and more blur. We also notice that due to the resolution of the pixelation the subject's left eye and eyebrow merge in a single brown block.

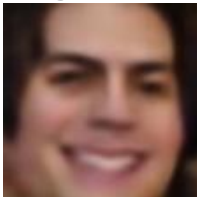
Base



Pixelated



CAE



CAE/MPL1



This image
Test Set

21.20 / 0.44
22.42 / 0.48

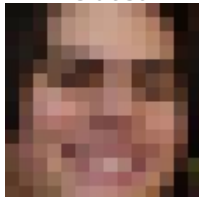
25.83 / 0.75
25.40 / 0.68

22.42 / 0.68
22.48 / 0.61

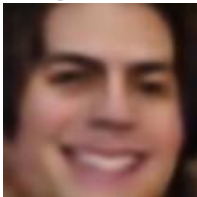
Base



Pixelated



CAE



CAE/MPL1



This image
Test Set

21.20 / 0.44
22.42 / 0.48

25.83 / 0.75
25.40 / 0.68

22.42 / 0.68
22.48 / 0.61

It can be noticed that the previously mentioned brown box is interpreted as a tilted eye.

This is actually somewhat correct: that single block is where an eye would usually be found, so it makes sense for the architecture to guess so.

This is actually somewhat correct: that single block is where an eye would usually be found, so it makes sense for the architecture to guess so.

Still this result shows that this architecture is clearly limited when it comes to stronger image tampering methods, and not even the perceptual loss can save the network from inaccurately reconstructing low-resolution pixelated inputs.

This is actually somewhat correct: that single block is where an eye would usually be found, so it makes sense for the architecture to guess so.

Still this result shows that this architecture is clearly limited when it comes to stronger image tampering methods, and not even the perceptual loss can save the network from inaccurately reconstructing low-resolution pixelated inputs.

This problem did not appear before because blurring is able to reduce detail while still maintaining the relative position of various facial features; with pixelation, instead, a lot of specific location details are lost within each pixelated block (e.g. the eye-eyebrow block previously discussed).

- 1 Introduction
- 2 Related Work
- 3 Dataset
- 4 Methods
 - Base Architecture
 - Improved Architectures
 - Loss Functions
- 5 Experiments
 - Architecture Experiments
 - Loss Function Experiments
 - Pixelation Experiments
- 6 Conclusions

The CAE generative architecture:

- is quite capable of reconstructing satisfactory approximations of the original image when it comes to gaussian blur image corruption;

The CAE generative architecture:

- is quite capable of reconstructing satisfactory approximations of the original image when it comes to gaussian blur image corruption;
- and can be considered relatively successful even in challenging situations (e.g. reconstructing the glasses in the Bill Gates image);

The CAE generative architecture:

- is quite capable of reconstructing satisfactory approximations of the original image when it comes to gaussian blur image corruption;
- and can be considered relatively successful even in challenging situations (e.g. reconstructing the glasses in the Bill Gates image);
- it somewhat struggles with strong pixelation, but reasonably so, as has been discussed.

- We found that simple modifications to the basic CAE architecture (e.g. adding variational/residual blocks) yield marginal performance improvements at best (at least with the ResCAE architecture; we were not able to successfully train a VAE to reach the same performance level).

- We found that simple modifications to the basic CAE architecture (e.g. adding variational/residual blocks) yield marginal performance improvements at best (at least with the ResCAE architecture; we were not able to successfully train a VAE to reach the same performance level).
- A significant boost in perceived reconstruction quality, as well as in min. number of epochs needed to reach satisfactory performance can be obtained by switching from the pixel loss to a perceptual one as described above. By combining the strengths of the two losses convergence can be further accelerated.

- We found that simple modifications to the basic CAE architecture (e.g. adding variational/residual blocks) yield marginal performance improvements at best (at least with the ResCAE architecture; we were not able to successfully train a VAE to reach the same performance level).
- A significant boost in perceived reconstruction quality, as well as in min. number of epochs needed to reach satisfactory performance can be obtained by switching from the pixel loss to a perceptual one as described above. By combining the strengths of the two losses convergence can be further accelerated.
- Future studies may explore what happens by performing different combinations of architecture parameters, e.g. by using the perceptual loss with the variational / resblocks CAE.

- As discussed above it's well established in the literature that PSNR / SSIM do not necessarily imply aesthetically pleasing reconstruction.

- As discussed above it's well established in the literature that PSNR / SSIM do not necessarily imply aesthetically pleasing reconstruction.
- We were able to reproduce this result, since in every example shown the best reconstructions also had the worst scores (except for the VAE results, which have bad scores since as discussed its training was unsuccessful).

- As discussed above it's well established in the literature that PSNR / SSIM do not necessarily imply aesthetically pleasing reconstruction.
- We were able to reproduce this result, since in every example shown the best reconstructions also had the worst scores (except for the VAE results, which have bad scores since as discussed its training was unsuccessful).
- This poses a problem to be tackled in future studies: how can one define a *quantitative* metric that still agrees with human intuition? This may be an interesting open problem.

We finally conclude by remarking that our study shows that (especially with limited computational resources) the denoising CAE can be an effective alternative to more expensive, state of the art solutions such as GAN-based architectures.

Thanks for your attention!

- [1] M. Ghanbari. “Scope of validity of PSNR in image/video quality assessment”. English. In: *Electronics Letters* 44 (13 June 2008), 800–801(1). ISSN: 0013-5194. URL: https://digital-library.theiet.org/content/journals/10.1049/el_20080522.
- [2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [3] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. “Transforming Auto-Encoders”. In: (2011). Ed. by Timo Honkela et al., pp. 44–51.
- [4] Gary B. Huang et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, Oct. 2007.

- [5] Q. Huynh-Thu and M. Ghanbari. “The accuracy of PSNR in predicting video quality for different video scenes and frame rates”. In: *Telecommun. Syst.* (49 2012), pp. 35–48. DOI: <https://doi.org/10.1007/s11235-010-9351-x>.
- [6] Daniel Jiwoong Im et al. “Denoising Criterion for Variational Auto-Encoding Framework”. In: *CoRR abs/1511.06406* (2015). arXiv: 1511.06406. URL: <http://arxiv.org/abs/1511.06406>.
- [7] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. In: *CoRR abs/1603.08155* (2016). arXiv: 1603.08155. URL: <http://arxiv.org/abs/1603.08155>.

- [8] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. “Accurate Image Super-Resolution Using Very Deep Convolutional Networks”. In: *CoRR* abs/1511.04587 (2015). arXiv: 1511.04587. URL: <http://arxiv.org/abs/1511.04587>.
- [9] Debarati Kundu and Brian L. Evans. “Full-reference visual quality assessment for synthetic images: A subjective study”. In: (2015), pp. 2374–2378. DOI: 10.1109/ICIP.2015.7351227.
- [10] Richard McPherson, Reza Shokri, and Vitaly Shmatikov. “Defeating Image Obfuscation with Deep Learning”. In: *CoRR* abs/1609.00408 (2016). arXiv: 1609.00408. URL: <http://arxiv.org/abs/1609.00408>.

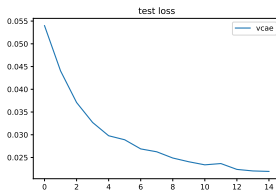
- [11] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. “Deep Face Recognition”. In: (Sept. 2015). Ed. by Mark W. Jones Xianghua Xie and Gary K. L. Tam, pp. 41.1–41.12. DOI: 10.5244/C.29.41. URL: <https://dx.doi.org/10.5244/C.29.41>.
- [12] H.R. Sheikh, M.F. Sabir, and A.C. Bovik. “A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms”. In: *IEEE Transactions on Image Processing* 15.11 (2006), pp. 3440–3451. DOI: 10.1109/TIP.2006.881959.
- [13] Jacob Conrad Trinidad. “Reconstructing Obfuscated Human Faces”. In: (). URL: <http://cs231n.stanford.edu/reports/2017/pdfs/223.pdf>.

- [14] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.
- [15] Erjin Zhou, Zhimin Cao, and Qi Yin. “Naive-Deep Face Recognition: Touching the Limit of LFW Benchmark or Not?” In: *CoRR* abs/1501.04690 (2015). arXiv: 1501.04690. URL: <http://arxiv.org/abs/1501.04690>.

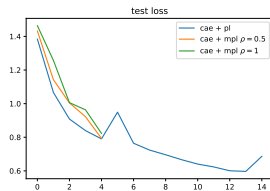
Appendix: Test Loss vs time



(a) CAE + ResCAE



(b) VCAE



(c) Perc. loss

Figure: Test loss against time for the final trained architectures.